МІНІСТЕРСТВО ОСВІТИ І НАУКИ

ХАРКІВСЬКИЙ КОМП′ЮТЕРНО-ТЕХНОЛОГІЧНИЙ КОЛЕДЖ

НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

**ЗВІТ**

**ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 2**

з дисципліни «Соціальні мережі»

|  |  |
| --- | --- |
| Виконав  студент гр. ОПК-315  Давидов Денис | Перевірив  викладач  Фесенко Д.В. |

Харків 2017**ЛАБОРАТОРНА РОБОТА №2**

«Основи галуження у Git. Робота з гілками»

**1. Тема роботи:** Основи використання галуження при створенні проектів.

**2 . Мета роботи:** Опанувати навичками роботи з гілками.

**Хід роботи:**

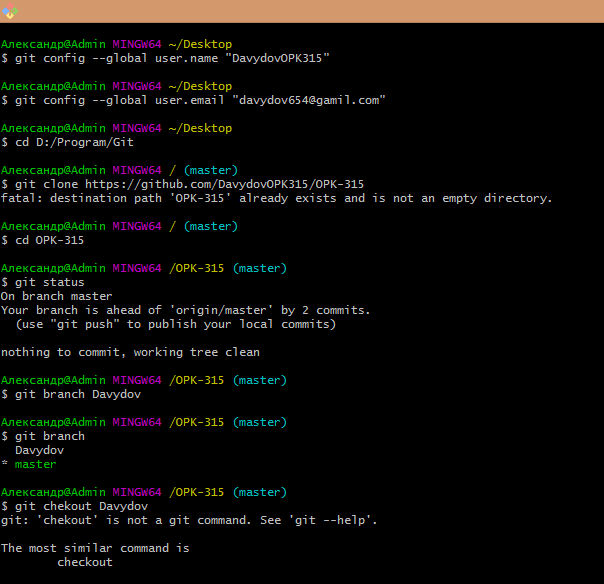


Рисунок 1.1 – Створення гілки та перехід на неї

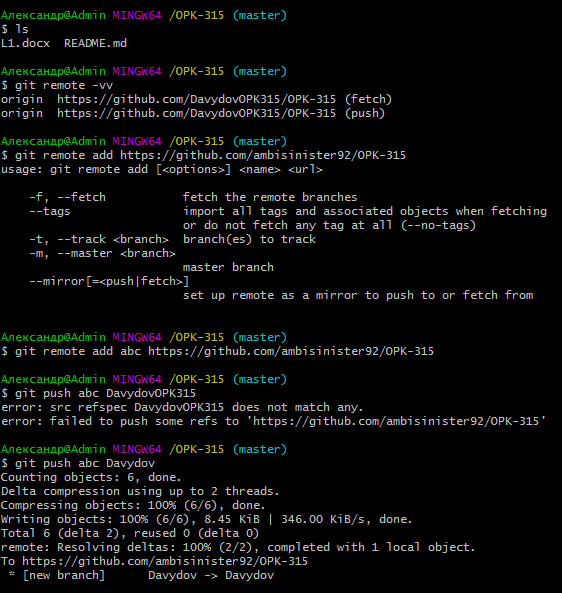


Рисунок 1.2 – Додання нового віддаленого репозиторію та злиття гілок

Код

#include <iostream> // библиотека ввода/вывода

#include <vector> // библиотека для подключения векторов

#include <algorithm> // библиотека для подключения алгоритмов

#include <time.h> // библиотека для процессорного времени

#include <math.h> // библиотека для основных алгебраических функций

using namespace std; //пространство имён

bool Proverka(vector<int> li, vector<int> ri) // проверка

{

if (li[0]<ri[0])

return true;

else if (li[0]>ri[0])

return false;

return li[1]>ri[1];

}

int main()

{

setlocale(LC\_ALL, "Russian");

srand(unsigned(time(NULL))); // для псевдослучайных чисел

int n;

n = rand() % (210 - 1) + 1; // подсчёт телевизоров

cout << n << "\n"; // вывод количества телевизоров

vector<vector<int>> mas(n, vector<int>(3)); // построение векторов

for (int i = 0; i<n; i++)

{

do

{

mas[i][0] = rand() % (109 - 1); // заполение массива

mas[i][1] = rand() % (109 - 1);

} while (mas[i][0]>mas[i][1]); // условие выхода

cout << i + 1 << ") ";

cout << mas[i][0] << " ";

cout << mas[i][1] << "\n";

mas[i][2] = i + 1;

}

sort(mas.begin(), mas.end(), Proverka); // вызов функции для проверки

cout << "\n";

for (int i = 0; i<n; i++) // цикл для проверки лишнего телевизора

{

int j = i + 1;

while (j<n && mas[j][0] <= mas[i][1])

{

if (mas[j][0] == mas[i][0] && mas[j][1] >= mas[i][1])

{

cout <<"Телевизор под номером "<<mas[i][2]<< " - это лишний телевизор\n";

system("Pause");

return 0;

}

mas[j][0] = mas[i][1] + 1;

if (mas[j][1] - mas[j][0]<0)

{

cout <<"Телевизор под номером "<<mas[j][2] << " - это лишний телевизор\n";

system("Pause");

return 0;

}

j++;

}

}

cout << -1; // если нет лишнего телевизора

system("Pause");

}

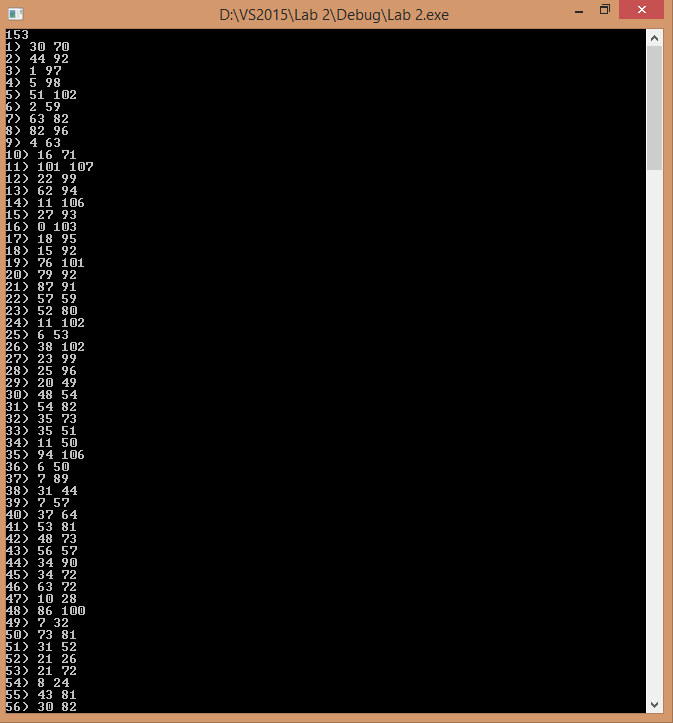


Рисунок 1.3 – загальна кількість телевізорів та їхні інтервали часу роботи

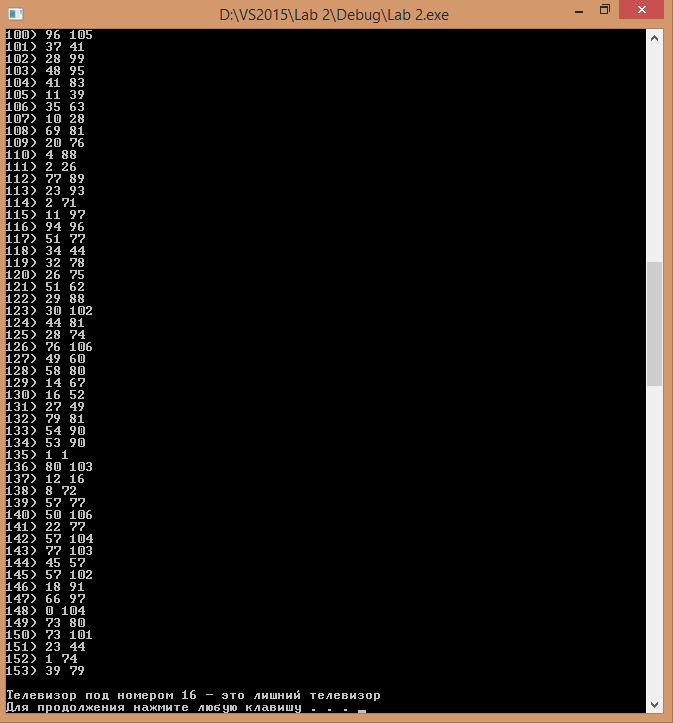


Рисунок 1.4 – інформація про зайвий телевізор

**Питання до захисту:**

1. **Чим є насправді гілки у Git?**

Гілка в Гіт це просто легкий двигаємий вказівник, що може пересуватися на одну з фіксацій (створених за допомогою git commit). Створення нової гілки створює новий вказівник, щоб можна було пересуватися. Особливий вказівник під назвою HEAD вказує на локальну гілку, на якій ви знаходитесь.

1. **Розкажіть про способи злиття у Git?**

Злиття буває двох типів: трьохпозиційне – злиття між двома останніми знімками станів з гілок (C3 і C4) і останнім спільним предком цих двох гілок (C2), створюючи новий знімок стану (і комміт), як показано на рисунку 1.5;

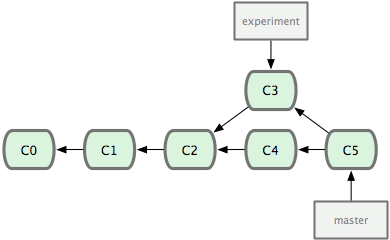


Рисунок 1.5 – трьохпозиційне злиття

злиття-перемотка (`fast-forward`) – git просто пересуває вказівник вперед, таким чином вказуючи на той же коміт що й гілка style. Але це можливо в історії без розгалужень.

Злиття змін гілки style в гілку master.

Style master

С1 <- C2 <- C3

1. **Що відбувається у робочій директорії при перемиканні гілок?**

При перемиканні гілки, файли у робочій папці змінюються. Якщо переключитися до старшої гілки, робоча папка буде повернута до того стану, який був на момент останнього фіксування у тій гілці. Якщо Гіт не зможе це зробити без проблем, він не дасть переключитися взагалі.

Роботу виконав: Давидов Д.О. Роботу перевірив: Фесенко Д.В.